

## Adv DevOps Lab Exp 04

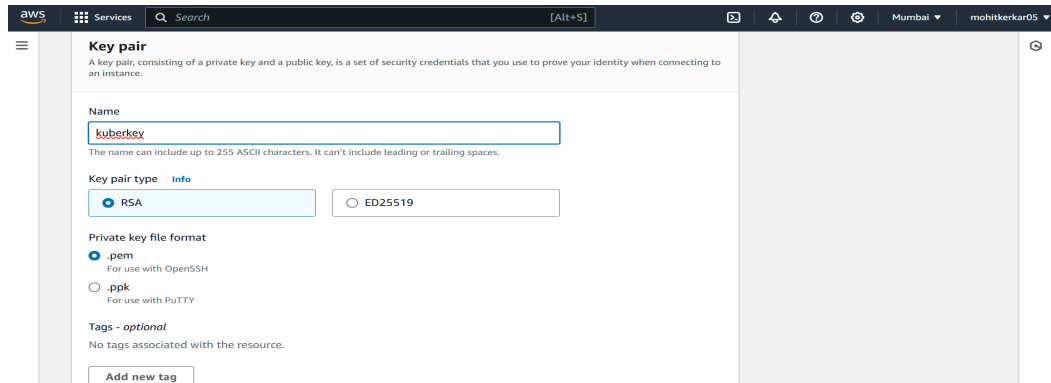
**Aim: To install Kubectl and execute Kubectl commands to manage the Kubernetes cluster and**

**deploy Your First Kubernetes Application.**

**(I performed this Experiment on my personal AWS account)**

### **Step 1: Creation of EC2 Instance, its Key-pair, remote login using SSH**

Firstly, create a key pair with a desired name and type 'RSA' with .pem extension. (.pem is useful with OpenSSH)

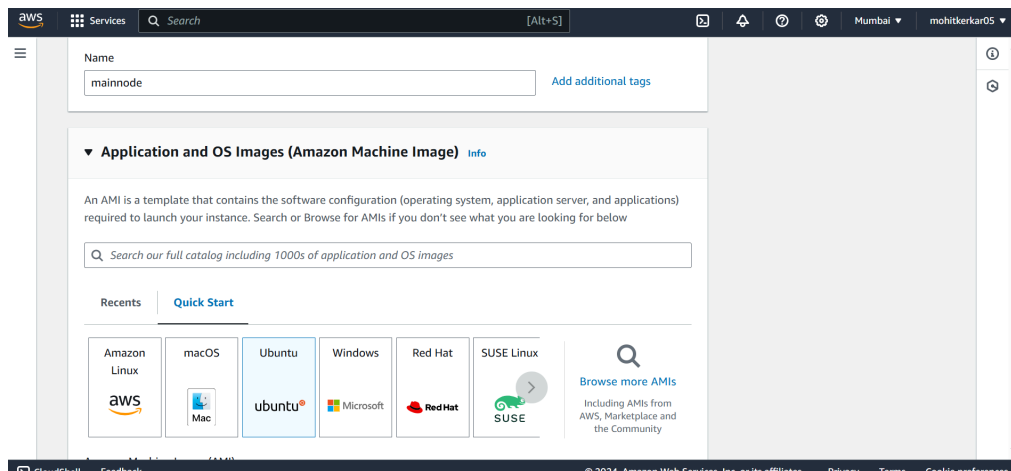


After creating the key, it automatically gets downloaded onto your machine. We are to create a new folder and place our within there. After this, we are to access our key and establish our connection with our instance by changing the directory to our key's directory

Now, proceed to the creation of the EC2 instance. I selected Amazon machine image as **Ubuntu** and instance type as **t2.medium**, just for the purpose of convenience and also for successful execution of the experiment.

The execution of the experiment is determined by the fact that how many resources is the instance allowed to access or use. The Free Tier eligible instance type **t2.micro** avails the instance with only one CPU, whereas with t2.medium we get 2 CPU and much more execution space.

Be mindful that along with all this, **t2.medium does not have free tier eligibility**. So, use the instances and the resources that you allow them to access resourcefully and economically. Shut/terminate them as and when work related to them is over



**▼ Instance type** [Info](#) | [Get advice](#)

**Instance type**  
**t2.medium**  
Family: t2 2 vCPU 4 GiB Memory Current generation: true  
On-Demand Linux base pricing: 0.0496 USD per Hour  
On-Demand Windows base pricing: 0.0676 USD per Hour  
On-Demand RHEL base pricing: 0.0784 USD per Hour  
On-Demand SUSE base pricing: 0.1496 USD per Hour

☐ AI  
[Compa](#)

Additional costs apply for AMIs with pre-installed software

Instances (1/1) <a href="#">Info</a>						
Last updated less than a minute ago <a href="#">Refresh</a> <a href="#">Connect</a> <a href="#">Instance state</a> <a href="#">Actions</a> <a href="#">Launch instances</a>						
<input type="text" value="Find Instance by attribute or tag (case-sensitive)"/>				<a href="#">All states</a>		
<input checked="" type="checkbox"/>	Name <a href="#">✎</a>	Instance ID	Instance state	Instance type	Status check	Alarm status
<input checked="" type="checkbox"/>	mainnode	i-0765530c8eaaa84cc	<span>Running</span>	t2.medium	Initializing	<a href="#">View alarms</a>

Here are my instance details. From here on, after remotely logging in from my local terminal, you would see my IPv4 address in my terminal (Powershell)

**i-0765530c8eaaa84cc (mainnode)**

i-0765530c8eaaa84cc (mainnode)

IPv6 address

—

Hostname type

IP name: ip-172-31-40-244.ap-south-1.compute.internal

35.154.201.225 | [open address](#)

Instance state

Running

Private IP DNS name (IPv4 only)

172.31.40.244

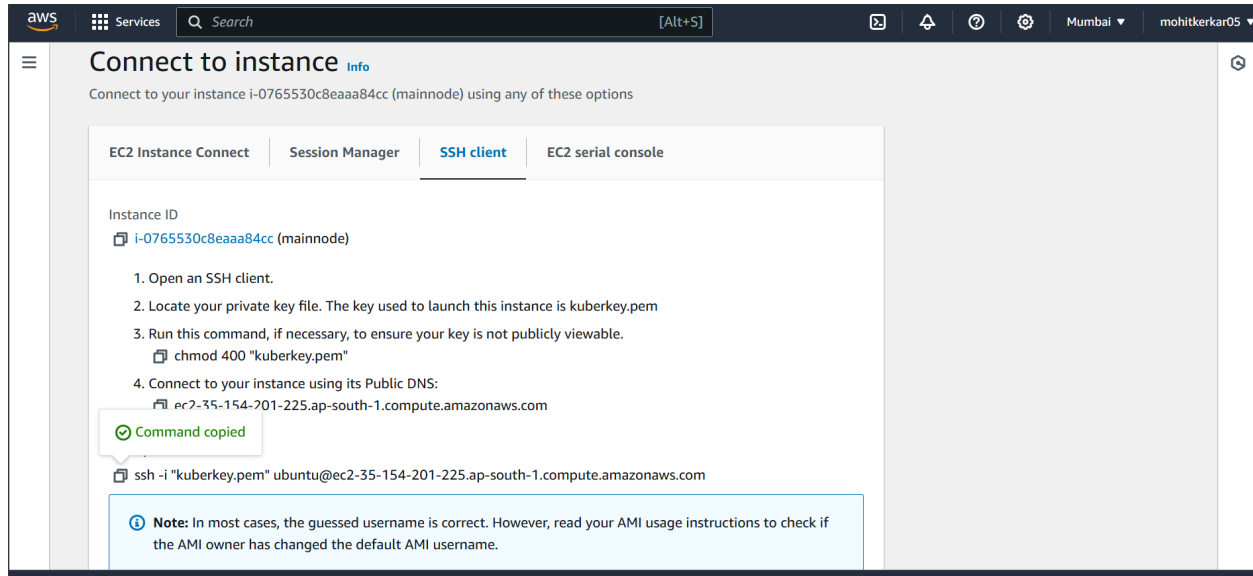
172.31.40.244

Public IPv4 DNS

ec2-35-154-201-225.ap-south-1.compute.amazonaws.com

[| open address](#)

Select the instance and press connect. Navigate to the SSH client section and copy the SSH command provided to us. We will require this for remotely logging into our instance from our local terminal.



As one can see, I changed my directory to the folder in which i had stored my key (.pem) and ran the ssh command so as to remotely login onto my instance and perform the necessary tasks from my terminal itself

```
ubuntu@ip-172-31-40-244: ~
Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Windows\system32> cd C:\Users\Dell\Desktop\keypair
PS C:\Users\Dell\Desktop\keypair> ssh -i "kuberkey.pem" ubuntu@ec2-35-154-201-225.ap-south-1.compute.amazonaws.com
The authenticity of host 'ec2-35-154-201-225.ap-south-1.compute.amazonaws.com (35.154.201.225)' can't be established.
ECDSA key fingerprint is SHA256:GIs2bs4t0fWY44Hiy3aN3Li34BT8eDD+LxwJMAdR7HU.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-35-154-201-225.ap-south-1.compute.amazonaws.com,35.154.201.225' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

System information as of Wed Sep 25 16:45:41 UTC 2024

System load:  0.0          Processes:      113
Usage of /:   22.8% of 6.71GB Users logged in: 0
Memory usage: 5%          IPv4 address for enX0: 172.31.40.244
Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.
```

## Step 2: Docker installation

Run the below mentioned commands in order to install docker on your instance

**Adding key:** `curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -`

**Adding key without apt-key:** `curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee /etc/apt/trusted.gpg.d/docker.gpg > /dev/null`

**Add docker repository:** `sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"`

```
ubuntu@ip-172-31-40-244:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
OK
ubuntu@ip-172-31-40-244:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee /etc/apt/trusted.gpg.d/docker.gpg > /dev/null
ubuntu@ip-172-31-40-244:~$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
Repository: 'deb [arch=amd64] https://download.docker.com/linux/ubuntu noble stable'
Description:
Archive for codename: noble components: stable
More info: https://download.docker.com/linux/ubuntu
Adding repository.
Press [ENTER] to continue or Ctrl-C to cancel.
Adding deb entry to /etc/apt/sources.list.d/archive_uri-https_download_docker_com_linux_ubuntu-noble.list
Adding disabled deb-src entry to /etc/apt/sources.list.d/archive_uri-https_download_docker_com_linux_ubuntu-noble.list
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 https://download.docker.com/linux/ubuntu noble InRelease [48.8 kB]
Get:5 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:6 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:7 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:8 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:9 https://download.docker.com/linux/ubuntu noble/stable amd64 Packages [15.3 kB]
Get:10 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [301 kB]
```

Run these commands to update the package list and install Docker:

`sudo apt update`

`sudo apt install docker-ce docker-ce-cli containerd.io`

```
ubuntu@ip-172-31-40-244:~$ sudo apt install docker-ce docker-ce-cli containerd.io
Setting up docker-compose-plugin (2.29.7-1~ubuntu.24.04~noble) ...
Setting up libltdl7:amd64 (2.4.7-7build1) ...
Setting up docker-ce-cli (5:27.3.1-1~ubuntu.24.04~noble) ...
Setting up libslirp0:amd64 (4.7.0-1ubuntu3) ...
Setting up pigz (2.8-1) ...
Setting up docker-ce-rootless-extras (5:27.3.1-1~ubuntu.24.04~noble) ...
Setting up slirp4netns (1.2.1-1build2) ...
Setting up docker-ce (5:27.3.1-1~ubuntu.24.04~noble) ...
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /usr/lib/systemd/system/docker.socket.
Processing triggers for man-db (2.12.0-4build2) ...
Processing triggers for libc-bin (2.39-0ubuntu8.2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-40-244:~$
```

Next, run this command....

`sudo mkdir -p /etc/docker`

`cat <<EOF | sudo tee /etc/docker/daemon.json`

```
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
EOF
```

```
sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker
```

```
ubuntu@ip-172-31-40-244: $ sudo mkdir -p /etc/docker
ubuntu@ip-172-31-40-244: $ cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
EOF
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
ubuntu@ip-172-31-40-244: $ sudo systemctl enable docker
Synchronizing state of docker.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable docker
ubuntu@ip-172-31-40-244: $ sudo systemctl daemon-reload
ubuntu@ip-172-31-40-244: $ sudo systemctl restart docker
```

### Step 3: Kubernetes Installation

Run these commands to install kubernetes:

```
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o
/etc/apt/keyrings/kubernetes-apt-keyring.gpg
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
```

```
ubuntu@ip-172-31-40-244: $ curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dear
mor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
ubuntu@ip-172-31-40-244: $ echo "deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.
io/core:/stable:/v1.31/deb/ /" | Out-File -FilePath "C:\path\to\your\directory\kubernetes.list" -Encoding utf8
Out-File: command not found
ubuntu@ip-172-31-40-244: $ echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
> https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /
ubuntu@ip-172-31-40-244: $
```

After this step, run the following commands,

```
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
```

Over here, while running `sudo apt-get update` if you encounter an error like this

```
ubuntu@ip-172-31-40-244:~$ sudo apt-get update
E: Malformed entry 1 in list file /etc/apt/sources.list.d/kubernetes.list (URI)
E: The list of sources could not be read.
```

Then run the following command and ensure that the file contains the following texts:

```
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /
```

```
ubuntu@ip-172-31-40-244: ~  
GNU nano 7.2 /etc/apt/sources.list.d/kubernetes.list  
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /
```

Be sure to save this file and then proceed with these commands

`sudo apt-get update`

`sudo apt-get install -y kubelet kubeadm kubectl`

`sudo apt-mark hold kubelet kubeadm kubectl`

```
ubuntu@ip-172-31-40-244: ~  
ubuntu@ip-172-31-40-244:~$ sudo apt-get update  
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease  
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]  
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease  
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease  
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease  
Get:6 https://prod-cdn.packages.k8s.io/repositories/iscv:/kubernetes:/core:/stable:/v1.31/deb InRelease [1186 B]  
Get:7 https://prod-cdn.packages.k8s.io/repositories/iscv:/kubernetes:/core:/stable:/v1.31/deb Packages [4865 B]  
Fetched 132 kB in 1s (199 kB/s)  
Reading package lists... Done  
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: The key(s) in the keyring /etc/apt/trusted.gpg.d/docker.gpg are ignored as the file has an unsupported filetype.  
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATION section in apt-key(8) for details.  
ubuntu@ip-172-31-40-244:~$ sudo apt-get install -y kubelet kubeadm kubectl  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
The following additional packages will be installed:  
  conntrack cri-tools kubernetes-cni  
The following NEW packages will be installed:  
  conntrack cri-tools kubeadm kubectl kubelet kubernetes-cni  
0 upgraded, 6 newly installed, 0 to remove and 142 not upgraded.  
Need to get 87.4 MB of archives.  
After this operation, 314 MB of additional disk space will be used.  
Get:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 conntrack amd64 1:1.4.8-1ubuntu1 [37.9 k
```

```
Setting up conntrack (1:1.4.8-1ubuntu1) ...  
Setting up kubectl (1.31.1-1.1) ...  
Setting up cri-tools (1.31.1-1.1) ...  
Setting up kubernetes-cni (1.5.1-1.1) ...  
Setting up kubeadm (1.31.1-1.1) ...  
Setting up kubelet (1.31.1-1.1) ...  
Processing triggers for man-db (2.12.0-4build2) ...  
Scanning processes...  
Scanning linux images...  
  
Running kernel seems to be up-to-date.  
  
No services need to be restarted.  
  
No containers need to be restarted.  
  
No user sessions are running outdated binaries.  
  
No VM guests are running outdated hypervisor (qemu) binaries on this host.  
ubuntu@ip-172-31-40-244:~$ sudo apt-mark hold kubelet kubeadm kubectl  
kubelet set on hold.  
kubeadm set on hold.  
kubectl set on hold.  
ubuntu@ip-172-31-40-244:~$
```

sudo systemctl enable --now kubelet

sudo kubeadm init --pod-network-cidr=10.244.0.0/16

```
ubuntu@ip-172-31-40-244:~$ sudo systemctl enable --now kubelet
ubuntu@ip-172-31-40-244:~$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
W0925 17:22:15.942659 6444 checks.go:1080] [preflight] WARNING: Couldn't create the interface used for talking to the container runtime: failed to create new CRI runtime service: validate service connection: validate CRI v1 runtime API for endpoint "unix:///var/run/containerd/containerd.sock": rpc error: code = Unimplemented desc = unknown service runtime.v1.RuntimeService
[WARNING FileExisting-socat]: socat not found in system path
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action beforehand using 'kubeadm config images pull'
error execution phase preflight: [preflight] Some fatal errors occurred:
failed to create new CRI runtime service: validate service connection: validate CRI v1 runtime API for endpoint "unix:///var/run/containerd/containerd.sock": rpc error: code = Unimplemented desc = unknown service runtime.v1.RuntimeService[preflight] If you know what you are doing, you can make a check non-fatal with '--ignore-preflight-errors=...'
To see the stack trace of this error execute with --v=5 or higher
```

Since running that command ran us into an error, we are expected to install containerd first

Run this command: sudo apt-get install -y containerd

Then run,

sudo mkdir -p /etc/containerd

sudo containerd config default | sudo tee /etc/containerd/config.toml

```
ubuntu@ip-172-31-40-244:~$ sudo apt-get update
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb InRelease
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: The key(s) in the keyring /etc/apt/trusted.gpg.d/docker.gpg are ignored as the file has an unsupported filetype.
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATION section in apt-key(8) for details.
ubuntu@ip-172-31-40-244:~$ sudo apt-get install -y containerd
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  docker-ce-rootless-extras libltdl7 libslirp0 pigz slirp4netns
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  runc
The following packages will be REMOVED:
  containerd.io docker-ce
The following NEW packages will be installed:
  containerd runc
0 upgraded, 2 newly installed, 2 to remove and 142 not upgraded.
Need to get 47.2 MB of archives.
```

Post this step, run these commands one by one

sudo systemctl restart containerd

sudo systemctl enable containerd

sudo systemctl status containerd

```

ubuntu@ip-172-31-40-244:~$ sudo systemctl restart containerd
ubuntu@ip-172-31-40-244:~$ sudo systemctl enable containerd
ubuntu@ip-172-31-40-244:~$ sudo systemctl status containerd
● containerd.service - containerd container runtime
   Loaded: loaded (/usr/lib/systemd/system/containerd.service; enabled; preset: enabled)
   Active: active (running) since Wed 2024-09-25 17:23:54 UTC; 17s ago
     Docs: https://containerd.io
   Main PID: 6555 (containerd)
    Tasks: 7
   Memory: 13.5M (peak: 14.1M)
      CPU: 121ms
   CGroup: /system.slice/containerd.service
           └─6555 /usr/bin/containerd

Sep 25 17:23:54 ip-172-31-40-244 containerd[6555]: time="2024-09-25T17:23:54.231856399Z" level=info msg="Start"
Sep 25 17:23:54 ip-172-31-40-244 containerd[6555]: time="2024-09-25T17:23:54.231887455Z" level=info msg="servin"
Sep 25 17:23:54 ip-172-31-40-244 containerd[6555]: time="2024-09-25T17:23:54.231896683Z" level=info msg="Start"
Sep 25 17:23:54 ip-172-31-40-244 containerd[6555]: time="2024-09-25T17:23:54.231919602Z" level=info msg="servin"
Sep 25 17:23:54 ip-172-31-40-244 containerd[6555]: time="2024-09-25T17:23:54.231952908Z" level=info msg="Start"
Sep 25 17:23:54 ip-172-31-40-244 containerd[6555]: time="2024-09-25T17:23:54.231972572Z" level=info msg="Start"
Sep 25 17:23:54 ip-172-31-40-244 containerd[6555]: time="2024-09-25T17:23:54.231980403Z" level=info msg="Start"
Sep 25 17:23:54 ip-172-31-40-244 containerd[6555]: time="2024-09-25T17:23:54.231986601Z" level=info msg="Start"
Sep 25 17:23:54 ip-172-31-40-244 systemd[1]: Started containerd.service - containerd container runtime.
Sep 25 17:23:54 ip-172-31-40-244 containerd[6555]: time="2024-09-25T17:23:54.234185443Z" level=info msg="conta

```

After this, run

`sudo apt-get install -y socat`

```

ubuntu@ip-172-31-40-244:~$ sudo apt-get install -y socat
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  docker-ce-rootless-extras libltdl7 libslirp0 pigz slirp4netns
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  socat
0 upgraded, 1 newly installed, 0 to remove and 142 not upgraded.
Need to get 374 kB of archives.
After this operation, 1649 kB of additional disk space will be used.
Get:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 socat amd64 1.8.0.0-4build3 [374 kB]
Fetched 374 kB in 0s (16.8 MB/s)
Selecting previously unselected package socat.
(Reading database ... 68108 files and directories currently installed.)
Preparing to unpack .../socat_1.8.0.0-4build3_amd64.deb ...
Unpacking socat (1.8.0.0-4build3) ...
Setting up socat (1.8.0.0-4build3) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

```

#### Step 4: Initialize kubecoluster:

`Sudo kubeadm init --pod-network-cidr=10.244.0.0/16`



```
ubuntu@ip-172-31-40-244: ~  
ubuntu@ip-172-31-40-244: $ sudo kubeadm init --pod-network-cidr=10.244.0.0/16  
[init] Using Kubernetes version: v1.31.0  
[preflight] Running pre-flight checks  
[preflight] Pulling images required for setting up a Kubernetes cluster  
[preflight] This might take a minute or two, depending on the speed of your internet connection  
[preflight] You can also perform this action beforehand using 'kubeadm config images pull'  
W0925 17:31:39.850175 7057 checks.go:846] detected that the sandbox image "registry.k8s.io/pause:3.8" of the  
container runtime is inconsistent with that used by kubeadm. It is recommended to use "registry.k8s.io/pause:3.  
10" as the CRI sandbox image.  
[certs] Using certificateDir folder "/etc/kubernetes/pki"  
[certs] Generating "ca" certificate and key  
[certs] Generating "apiserver" certificate and key  
[certs] apiserver serving cert is signed for DNS names [ip-172-31-40-244 kubernet es.kubernet es.default kubernet  
es.default.svc kubernet es.default.svc.cluster.local] and IPs [10.96.0.1 172.31.40.244]  
[certs] Generating "apiserver-kubelet-client" certificate and key  
[certs] Generating "front-proxy-ca" certificate and key  
[certs] Generating "front-proxy-client" certificate and key  
[certs] Generating "etcd/ca" certificate and key  
[certs] Generating "etcd/server" certificate and key  
[certs] etcd/server serving cert is signed for DNS names [ip-172-31-40-244 localhost] and IPs [172.31.40.244 12  
7.0.0.1 ::1]  
[certs] Generating "etcd/peer" certificate and key  
[certs] etcd/peer serving cert is signed for DNS names [ip-172-31-40-244 localhost] and IPs [172.31.40.244 127.  
0.0.1 ::1]  
[certs] Generating "etcd/healthcheck-client" certificate and key  
[certs] Generating "apiserver-etcd-client" certificate and key  
[certs] Generating "sa" key and public key  
  
[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public" namespace  
[kubelet-finalize] Updating "/etc/kubernetes/kubelet.conf" to point to a rotatable kubelet client certificate a  
nd key  
[addons] Applied essential addon: CoreDNS  
[addons] Applied essential addon: kube-proxy  
  
Your Kubernetes control-plane has initialized successfully!  
  
To start using your cluster, you need to run the following as a regular user:  
  
    mkdir -p $HOME/.kube  
    sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config  
    sudo chown $(id -u):$(id -g) $HOME/.kube/config  
  
Alternatively, if you are the root user, you can run:  
  
    export KUBECONFIG=/etc/kubernetes/admin.conf  
  
You should now deploy a pod network to the cluster.  
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:  
    https://kubernetes.io/docs/concepts/cluster-administration/addons/  
  
Then you can join any number of worker nodes by running the following on each as root:  
  
kubeadm join 172.31.40.244:6443 --token egcpta.ggcdfxh328wg5vm3 \  
--discovery-token-ca-cert-hash sha256:e392dc659374560eb409e01b69491d214ea39cb415ee934aaecfcff6a7f21d5  
ubuntu@ip-172-31-40-244: $
```

Copy the mkdir and chown commands from the top and execute them.

```
mkdir -p $HOME/.kube
```

```
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
```

```
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Add a common networking plugin called flannel as mentioned in the code.

kubectl apply -f

<https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml>

```
ubuntu@ip-172-31-40-244:~$ kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
namespace/kube-flannel created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
ubuntu@ip-172-31-40-244:~$
```

Now that the cluster is up and running, we can deploy our nginx server on this cluster. Apply this deployment file using this command to create a deployment

kubectl apply -f <https://k8s.io/examples/application/deployment.yaml>

```
ubuntu@ip-172-31-40-244:~$ kubectl apply -f https://k8s.io/examples/application/deployment.yaml
deployment.apps/nginx-deployment created
ubuntu@ip-172-31-40-244:~$
```

kubectl get pods

```
ubuntu@ip-172-31-40-244:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-brgb7   0/1     Pending   0           48s
nginx-deployment-d556bf558-s5rtj   0/1     Pending   0           48s
ubuntu@ip-172-31-40-244:~$
```

POD\_NAME=\$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")

kubectl port-forward \$POD\_NAME 8080:80

```
ubuntu@ip-172-31-40-244:~$ POD_NAME=$(kubectl get pods -l app=nginx -o jsonpath="{.items[0].metadata.name}")
ubuntu@ip-172-31-40-244:~$ kubectl port-forward $POD_NAME 8080:80
error: unable to forward port because pod is not running. Current status=Pending
ubuntu@ip-172-31-40-244:~$
```

To untaint all nodes, run the following command

kubectl taint nodes --all [node-role.kubernetes.io/control-plane:NoSchedule-](https://kubernetes.io/control-plane/NoSchedule-)

Then run, kubectl get nodes

```
ubuntu@ip-172-31-40-244:~$ kubectl taint nodes --all node-role.kubernetes.io/control-plane:NoSchedule-
node/ip-172-31-40-244 untainted
ubuntu@ip-172-31-40-244:~$ kubectl get nodes
NAME                STATUS    ROLES    AGE   VERSION
ip-172-31-40-244    Ready    control-plane   16m   v1.31.1
ubuntu@ip-172-31-40-244:~$
```

Then run, kubectl get pods command again

```
ubuntu@ip-172-31-40-244:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-brgb7    1/1     Running   0           7m15s
nginx-deployment-d556bf558-s5rtj    1/1     Running   0           7m15s
ubuntu@ip-172-31-40-244:~$
```

Since, Jenkins server/service is already occupying port 8080, i tried starting the nginx service at 8082 port number. Following is the command that i executed was,  
kubectl port-forward \$POD\_NAME 8082:80

```
ubuntu@ip-172-31-40-244:~$ kubectl port-forward svc/nginx 8082:80
Forwarding from 127.0.0.1:8082 -> 80
Forwarding from [::1]:8082 -> 80
Handling connection for 8082
```

The last step consisted of opening a new terminal and logging in using the SSH method that we used earlier and running the following command so as to finally confirm that we have successfully deployed our kubernetes application over the nginx server, using nginx service (optional)

```
curl --head http://127.0.0.1:8082
```

```
Last login: Wed Sep 25 19:13:52 2024 from 45.112.58.76
ubuntu@ip-172-31-40-244:~$ curl --head http://127.0.0.1:8082
HTTP/1.1 200 OK
Server: nginx/1.14.2
Date: Wed, 25 Sep 2024 19:36:26 GMT
Content-Type: text/html
Content-Length: 612
Last-Modified: Tue, 04 Dec 2018 14:44:49 GMT
Connection: keep-alive
ETag: "5c0692e1-264"
Accept-Ranges: bytes
```

#### Conclusion:

In this experiment, we successfully deployed an Nginx web server on a Kubernetes cluster and exposed it using NodePort services. By creating the deployment and configuring the service, we demonstrated Kubernetes' ability to manage containerized applications efficiently.

The use of 'kubectl' commands allowed us to check pod status and access the Nginx server locally via port forwarding on the EC2 instance. This experience provided valuable insights into service management and networking concepts within Kubernetes.

Overall, this experiment reinforced the fundamental skills needed for deploying cloud-native applications and laid the groundwork for exploring more advanced Kubernetes features in future projects.